2014

Automatic Curvilinear Quality Mesh Generation Driven by Smooth Boundary and Guaranteed Fidelity

Jing Xu
Old Dominion University

Andrey N. Chernikov
Old Dominion University

Original Publication Citation

Xu, J., & Chernikov, A. N. (2014). Automatic curvilinear quality mesh generation driven by smooth boundary and guaranteed fidelity. *Procedia Engineering, 82*(Supplement C), 200-212. doi: 10.1016/j.proeng.2014.10.384

Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 82 (2014) 200 – 212

Procedia Engineering

www.elsevier.com/locate/procedia

23rd International Meshing Roundtable (IMR23)

Automatic curvilinear quality mesh generation driven by smooth boundary and guaranteed fidelity

Jing Xu*, Andrey N. Chernikov

Old Dominion University, Department of Computer Science, 4700 Elkhorn Ave, Norfolk, VA, 23508, USA

Abstract

The development of robust high-order finite element methods requires the construction of valid high-order meshes for complex geometries without user intervention. This paper presents a novel approach for automatically generating a high-order mesh with two main features: first, the boundary of the mesh is globally smooth; second, the mesh boundary satisfies a required fidelity tolerance. Invalid elements are eliminated. Example meshes demonstrate the features of the algorithm.

Peer-review under responsibility of organizing committee of the 23rd International Meshing Roundtable (IMR23)
Keywords: mesh generation; high-order; Bézier polynomial basis; smoothness

1. Introduction

High-order finite element methods have been used extensively in direct numerical simulations in the last few decades. The exponential rates of convergence, small dispersion and diffusion solution errors have all motivated the development of higher-order finite element techniques which better capture the geometry [1,2,11]. How well the geometry is approximated has fundamentally important effects on the accuracy of finite element solutions [6,10]. Therefore, valid meshes with properly curved elements must be constructed to approximate the curved geometric domain.

The discretization error results from the fact that a function of a continuous variable is represented in the computer by a finite number of evaluations. In conventional meshes with all straight-sided elements, the discretization error is usually controlled by making sufficiently small elements where geometric features occur such as on the objects' boundary. But this is not numerically efficient in the sense that the cost of assembling and solving a sparse system of linear equations in the FE method directly depends on the number of elements. The high-order methods however, decompose the solution domain into fewer elemental regions that capture the features of the geometry.

There are two ways to accomplish the generation of a curvilinear mesh when a geometric domain is given. The first is to directly create a valid curvilinear boundary and interior discretization with required size and shape of the

*Corresponding author.
E-mail address: jxu@cs.odu.edu, achernik@cs.odu.edu

Peer-review under responsibility of organizing committee of the 23rd International Meshing Roundtable (IMR23)
doi:10.1016/j.proeng.2014.10.384

elements. The second way is to initially construct a straightedge discretization of the model geometry, followed by the transformation of that discretization into high-order elements suitable for a high-order FE method.

Various procedures have been developed and implemented using the latter approach. Sherwin and Peiro [19] presented a high-order unstructured mesh generation algorithm. A linear triangular surface mesh is first generated, the transformation of that mesh into high-order surface is performed, and finally a curved mesh is constructed of the interior volume. Three strategies are adopted to alleviate the problem of invalid high-order meshes: optimization of the surface mesh that accounts for surface curvature, hybrid meshing with prismatic elements near the domain boundaries and curvature driven surface mesh adaption.

Dey et al. [6] described an iterative algorithm for curving straight-edge meshes using quadratic Lagrange interpolation functions. First, all mesh edges and faces classified on curved model boundaries are curved. Second, the intersections between mesh edges on the model surface are detected and eliminated. Third, invalid curved mesh regions are corrected by using local mesh modification tools.

Shephard et al. [18] discussed the automatic generation of adaptively controlled meshes for general three-dimensional domains. The algorithm starts with isolating all of the edges and vertices in the model that will have singularities, constructing of a coarse linear mesh on the boundary of the model with appropriate geometric gradation towards the isolated singular features and constructing of a coarse linear mesh of the remainder of the domain. Then the algorithm curves the singular feature isolation mesh, and the remaining mesh entities classified on the curved boundaries. Finally, mesh modification is applied to ensure a valid mesh of acceptably shaped elements.

Luo et al. [15] isolates singular reentrant model entities, then generates linear elements around those features, and curves them while maintaining the gradation. Linear elements are generated for the rest of the domain, and those elements that are classified on the curved boundary, are transformed into curved elements conforming to the curved boundary. Modification operations are applied to eliminate invalid elements whenever they are introduced. Later, they extended their work to adapted boundary layer meshes to allow for higher-order analysis of viscous flows [17]. The layered structure of anisotropic elements in the boundary layer meshes is able to construct elements with proper configuration and gradation.

George and Borouchaki [8] proposed a method for constructing tetrahedral meshes of degree two from a polynomial surface mesh of degree two. Corresponding linear surface mesh is first extracted, followed by constructing the linear volumetric mesh. Next the algorithm enriches the linear mesh to the polynomial of degree two mesh by introducing the edge nodes. After that *Jacobian* is introduced for guiding the correction of the invalid curved elements. Finally an optimization procedure is used to enhance the quality of the curved mesh.

Lu et al. [13] presented a parallel mesh adaptation method with curved element geometry. The core of the algorithm is made up of two classes of mesh modification. Element invalidity and shape quality problems are resolved by curved entity reshaping operations and by local mesh modifications.

The validity of a curved mesh is crucial to the successful execution of high-order finite element simulations. To verify the validity, it is necessary to calculate the determinant of the Jacobian matrix (*Jacobian*). A curved element is valid if and only if its *Jacobian* is strictly positive everywhere. However, it is cumbersome to verify the element validity when Lagrangian polynomial is used because calculating *Jacobian* becomes computationally and geometrically complex. Prior work shows that the properties of Bézier polynomials provide an attractive solution [8,9,13,14]. A lower bound for the *Jacobian* can be evaluated by the convex hull property of the Bézier polynomials [7]. If the lower bound is not tight enough, either degree elevation procedure or subdivision procedure is selected to yield a tighter lower bound [8,13,14]. Johnen et al. [9] expands the *Jacobian* using Bézier polynomial basis. Based on its properties, boundedness and positivity were obtained to provide an efficient way to determine the validity and to measure the distortion.

In this paper, a new approach is proposed for automatically generating a high-order mesh to represent geometry with smooth mesh boundaries and graded interior with guaranteed fidelity. Cubic Bézier polynomial basis is selected for the geometric representation of the elements because it provides a convenient framework supporting the smooth operation while maintaining guaranteed fidelity. We list the contributions in this paper here. To our knowledge, no consideration was given to them in prior work.

- Curved mesh boundary is globally smooth, i.e., its tangent is everywhere continuous.
- Curved mesh boundary everywhere satisfies a user-defined fidelity tolerance.

The procedure starts with the automatic construction of a graded linear mesh that simultaneously satisfies the quality and the fidelity requirements. The edges of those linear elements which are classified on curved boundary are then curved using cubic Bézier polynomial basis while maintaining the smoothness. To resist inverted elements, the procedure next curves the interior elements by solving for the equilibrium configuration of an elasticity problem. A validity verification procedure demonstrates that intersection edges and highly distorted elements are eliminated.

The rest of the paper is organized as follows. In Section 2, we review some basic definitions and materials. Section 3 gives a description of the automatic construction of graded linear mesh, while Section 4 describes the transformation of those linear mesh elements into high-order elements. Section 5 presents the validity checking algorithms. Section 6 proves mesh fidelity. We present meshing results in Section 7 and conclude in Section 8.

2. Preliminaries

The method uses the cubic Bézier polynomial basis to construct a high-order mesh that has smooth boundaries. The idea is to deform the linear mesh edges such that the curved edges conform to the expected domain boundary. The determinant of the Jacobian matrix is used to determine the validity. In this section we review Bézier curves, Bézier triangles and the Jacobian.

2.1. Bézier curves

We will express Bézier curves in terms of Bernstein polynomials. A *nth* order Bernstein polynomial is defined explicitly by

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}, \quad i = 0, ..., n, \quad t \in [0, 1],$$

where the binomial coefficients are given by

$$\binom{n}{i} = \begin{cases} \frac{n!}{i!(n-i)!} & \text{if } 0 \le i \le n \\ 0 & \text{else.} \end{cases}$$

One of the important properties of the Bernstein polynomials is that they satisfy the following recursion:

$$B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t),$$

then a recursive definition for the Bézier curve of degree n expresses it as a point-to-point linear combination (linear interpolation) of a pair of corresponding points in two Bézier curves of degree $n-1$.

Given a set of points $P_0, P_1, ..., P_n \in E^3$, where E^3 is three-dimensional euclidean space, and $t \in [0, 1]$, set

$$b_i^r(t) = (1-t)b_i^{r-1}(t) + tb_{i+1}^{r-1}(t) \quad \begin{cases} r = 1, ..., n \\ i = 0, ..., n-r \end{cases}$$

and $b_i^0(t) = P_i$. Then $b_0^n(t)$ is the point with parameter value t on the Bézier curve b^n. The polygon P formed by $P_0, P_1, ..., P_n$ is called *control polygon* of the curve b^n, and the polygon vertices P_i are called *control points*.

An explicit form of a *nth* order Bézier curve can be defined as

$$b^n(t) = \sum_{i=0}^{n} B_i^n(t) P_i.$$

The barycentric form of Bézier curves can demonstrate its symmetry property nicely. Let u and v be the barycentric coordinates, $u \in [0, 1]$ and $v \in [0, 1]$, $u + v = 1$, then

$$b^n(u, v) = \sum_{i+j=n} B_{ij}^n(u, v) P_{ij},$$

where $P_{ij} \in E^3$ are the control points, and $i + j = n$.

Specifically, a cubic Bézier curve can be written in terms of the barycentric coordinates,

$$b^3(u,v) = \sum_{i+j=3} B^3_{ij}(u,v)P_{ij},$$

where $B^3_{ij}(u,v) = \frac{3!}{i!j!}u^i v^j$, $u \in [0,1]$ and $v \in [0,1]$ are the barycentric coordinates and $u+v=1$.

2.2. Bézier triangles

Univariate Bernstein polynomials are the terms of the binomial expansion of $[t+(1-t)]^n$. In the bivariate case, a *nth* order Bernstain polynomial is defined by

$$B^n_{\vec{i}}(\vec{u}) = \binom{n}{\vec{i}} u^i v^j w^k, \quad |\vec{i}| = n,$$

where $u \in [0,1]$, $v \in [0,1]$ and $w \in [0,1]$ are the barycentric coordinates and $u+v+w=1$. This follows standard convention for the *trinomial coefficients* $\binom{n}{\vec{i}} = \frac{n!}{i!j!k!}$.

This leads to a simple definition of a Bézier triangle of degree n

$$\mathcal{T}^n(u,v,w) = \sum_{i+j+k=n} B^n_{ijk}(u,v,w)P_{ijk},$$

where P_{ijk} are the control points. Specifically, a Bézier triangle of degree three can be written as

$$\mathcal{T}^3(u,v,w) = \sum_{i+j+k=3} B^3_{ijk}(u,v,w)P_{ijk},$$

where $B^3_{ijk}(u,v,w) = \frac{3!}{i!j!k!}u^i v^j w^k$, $u \in [0,1]$, $v \in [0,1]$ and $w \in [0,1]$ are the barycentric coordinates and $u+v+w=1$.

The Bézier mesh geometry shape possesses three important properties which are useful to this work:

- The Convex Hull Property: A Bézier curve, surface or volume is contained in the convex hull formed by its control points;
- All derivatives and products of Bézier functions are Bézier functions;
- The convex hull can be refined by Bézier degree elevation algorithm or Bézier subdivision algorithm.

2.3. The Jacobian

We explore the concept of a derivative of a coordinate transformation, which is known as the *Jacobian* of the transformation.

Let's start at the definition of a finite element. A typical finite element e, $e \in R^n$, is defined by a closed subset of K, $K \in R^n$ with a non empty interior, a set of real-valued functions N defined over the set K, and a finite set of local nodes $u_i, 1 \le i \le N$. Then the mapping from the set of local coordinates $\hat{x}$, $\hat{y}$ to a corresponding set of global coordinates x, y is:

$$\vec{u} = \sum_a N_a \vec{u}_a = [N_1, N_2, ..., N_N] \begin{bmatrix} u_1 \\ u_2 \\ ... \\ u_N \end{bmatrix}, \quad a = 1, 2, ..., N,$$

where $\vec{u} = u(x,y)$, and $\vec{u}_a = u_a(\hat{x},\hat{y})$. The functions $N_a, a = 1, 2, ..., N$ are called *shape functions* (or basis functions).

By the *chain rule* of partial differentiation we have

$$\begin{bmatrix} \frac{\partial N_a}{\partial \hat{x}} & \frac{\partial N_a}{\partial \hat{y}} \end{bmatrix} = \begin{bmatrix} \frac{\partial N_a}{\partial x} & \frac{\partial N_a}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial x}{\partial \hat{x}} & \frac{\partial x}{\partial \hat{y}} \\ \frac{\partial y}{\partial \hat{x}} & \frac{\partial y}{\partial \hat{y}} \end{bmatrix} = \begin{bmatrix} \frac{\partial N_a}{\partial x} & \frac{\partial N_a}{\partial y} \end{bmatrix} J,$$

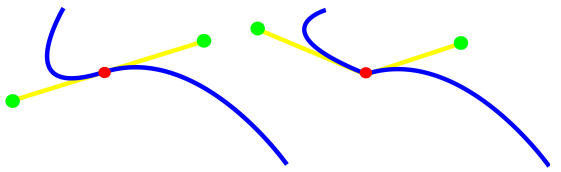


Fig. 1: Two Bézier paths with their control points (in green), made out of two cubic Bézier curves connected by the *endpoints* (in red). The yellow line segments are tangents to the both sides of the red Bézier *endpoint*. Left: a smooth Bézier path because the two green control points and the red *endpoint* lie in a straight line. Right: a Bézier path with a cusp where the curves connect, because the two green control points and the red *endpoint* do not lie in a straight line.

$$J = \begin{bmatrix} \frac{\partial x}{\partial \hat{x}} & \frac{\partial x}{\partial \hat{y}} \\ \frac{\partial y}{\partial \hat{x}} & \frac{\partial y}{\partial \hat{y}} \end{bmatrix}.$$

J is known as the *Jacobian matrix* for the transformation. As x, y are explicitly given by the relation defining the curvilinear coordinates, the matrix J can be found explicitly in terms of the local coordinates.

3. Linear mesh construction

We adopt the image-to-mesh conversion algorithm [3], for four reasons: (1) it allows for a guaranteed angle bound (quality), (2) it allows for a guaranteed bound on the distance between the boundaries of the mesh and the boundaries of the object (fidelity), (3) it coarsens the mesh to a much lower number of elements with gradation in the interior, (4) it is formulated to work in both two and three dimensions. Once we have a high quality linear mesh, we are about to construct curvilinear mesh based on it as the next step.

4. Curvilinear mesh transformation from linear meshes

Although it is attractive to construct valid high-order meshes by curving mesh entities classified on curved boundaries and the remainder of the domain simultaneously, in practice we transform the linear mesh entities classified on boundaries followed by curving mesh entities in the interior while eliminating invalid elements. Bézier curve basis is selected because its mathematical descriptions are compact, intuitive, and elegant. It is easy to compute, easy to use in higher dimensions (3D and up), and can be stitched together to represent any shape.

4.1. Constructing smooth Bézier paths from boundary mesh entities

A curve or surface can be described as having C^n continuity, n being the measure of smoothness. Consider the segments on either side of a point on a curve:

C^0: The curves touch at the joint point;

C^1: First derivatives are continuous;

C^2: First and second derivatives are continuous.

We aim to find a smooth C^1 curve passing through all the mesh boundary points given in order. A Bézier path is C^1 smooth provided that two Bézier curves share a common tangent direction at the join point. In other words, each endpoint and its two surrounding control points lie in a straight line. Fig. 1 shows two Bézier paths with their control points.

The basic idea is to calculate control points around each endpoint so that they lie in a straight line with the endpoint. However, curved segments would not flow smoothly together when quadratic Bézier form (three control points) is

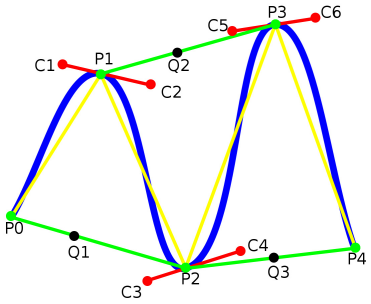


Fig. 2: An example of finding control points of a smooth cubic Bézier path. For the curve between P_1 and P_2, we need C_2 and C_3. On segment P_0P_2, find a point Q_1 such that$|P_0Q_1|/|Q_1P_2|= |P_0P_1|/|P_1P_2|$. Translate segment P_0P_2 so that point Q_1 lies on point P_1, and scale the length of translated segment P_0P_2, then the new position of point P_2 is the position of control point C_2. Similarly, the position of control point C_3 can be found by translating segment P_1P_3 such that point Q_2 lies on point P_2.

used. Instead, we need to go one order higher to a cubic Bézier (four control points) so we can build "S" shaped segments.

The points we have in hand are only endpoints of boundary segments, so the task becomes to find the other two control points to define the Bézier curve. We find these control points by translating the segments formed by the lines between the previous *endpoint* and the next *endpoint* such that these segments become the tangents of the curves at the *endpoints*. We scale these segments to control the curvature. An example is illustrated in Fig. 2.

4.2. Curving mesh entities in the interior

It is usually not enough to curve only the boundary mesh edges because self-intersecting mesh edges may appear which lead to invalid elements. In such cases, interior mesh elements should also be curved to eliminate the invalidity or improve curved element quality. Local mesh modifications such as minimizing the deformation, edge or facet deletion, splitting, collapsing, swapping as well as shape manipulation have been used to correct an invalid region [6, 8,13,15].

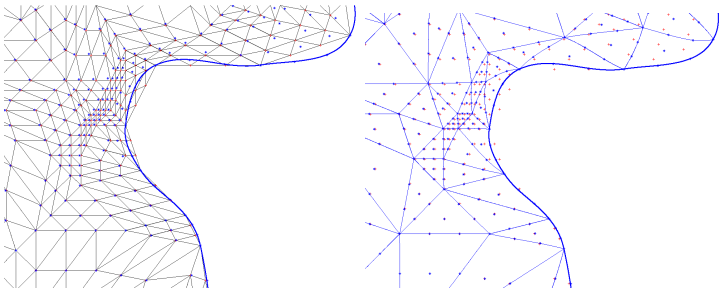
Persson and Peraire [16] proposed a node relocation strategy for constructing well-shaped curved mesh. They use a nonlinear elasticity analogy, where the geometry of the domain to be meshed is represented as an elastic solid. By solving for the equilibrium configuration, vertices located in the interior are relocated as a result of a prescribed boundary displacement. We will follow this idea in this section.

For each mesh edge, we find the positions of the two nodes that are located in the one-third and two-thirds ratio of each edge of the linear mesh. These positions are original positions of these nodes before deformation. We find the control points corresponding to the new positions of these nodes for the interior mesh edges after the mesh is deformed. We deform the mesh such that the two nodes on each boundary edge of the linear mesh move to the corresponding positions (one-third and two-thirds ratio) on the curved boundary edge. The vertices on boundary edges maintain their positions because they are already on the curved boundary. The new coordinates of all the vertices and nodes on the interior edges after deformation are computed by solving an elastic finite element problem [21]. As a result, the elements of the linear mesh are deformed minimally and proportionally to their distance to the points lying on the curved mesh boundary and to the amount of the displacement at these boundary vertices and nodes. Fig. 3 illustrates this step.

Using the new positions of these points after deformation, the corresponding control points that determine the curved edge passing through the points in the new positions can be easily calculated:

$$C_1 = -\frac{5}{6}v_0 + \frac{1}{3}v_1 + 3v_3 - \frac{3}{2}v_4,$$

$$C_2 = \frac{1}{3}v_0 - \frac{5}{6}v_1 - \frac{3}{2}v_3 + 3v_4,$$



(a) (b)

Fig. 3: An illustration of the solid mechanics approach to curved mesh generation. The bold blue line is curved boundary, the red crosses show the original vertex positions, the blue stars show the new positions after the elements are deformed according to the solution of a nonlinear elasticity problem, the gray segments show the displacements. (a) Before deformation, each one of the linear elements is divided into 9 sub-elements, with one-third and two-thirds ratio of each edge. (b) Elements are deformed according to the equilibrium solution of a linear elasticity problem.

where C_1 and C_2 are two middle control points of the four control points of the interior mesh edge, and v_0, v_1, v_2 and v_3 are points the curved edge passes through.

Validity check is executed after this procedure. In most cases, it can handle this problem successfully. However, in the case that the curvature of the boundary edges is very large, the interior linear edges may not be curved enough to avoid the intersection. Once our validity checking procedure reports that there is an invalid element, local mesh modifications can be used to correct the shape.

5. Element validity

A curvilinear mesh is valid provided that the intersection of the interiors of two different elements is the null set and any two mesh edges or faces do not intersect each other (except the common vertices or edges). To verify a curved element, we can use explicit intersection checks if the number of elements is small. A cheaper way is detecting the intersection at the element level by evaluating the sign of the *Jacobian* throughout the element. One approach is verifying the positiveness by sampling the *Jacobian* at discrete locations [12]. A more precise way is to calculate a lower bound for the *Jacobian*. When the Bézier form is used to map a reference element, the *Jacobian* is also a Bézier function with order $q = dimension * (degree - 1)$ [14], it is easy to be obtained due to its convex hull property [7]. In the case that a positive lower bound is obtained, it guarantees that the element is valid; on the contrary, when a non-positive bound occurs, the element may or may not be invalid. In this case we need to obtain a tighter bound. This evaluation can be either used to check the validity or to guide the correction of invalid elements.

We rewrite a cubic Bézier triangle $\mathcal{T}^3(u,v,w)$ in the following form:

$$\mathcal{T}^3(u,v,w) = P_{300}u^3 + P_{030}v^3 + P_{003}w^3 + 3P_{201}u^2w + 3P_{210}u^2v + 3P_{120}uv^2 + 3P_{102}uw^2 + 3P_{021}v^2w + 3P_{012}vw^2 + 6P_{111}uvw.$$

The Jacobian matrix of a Bézier triangle can be written as

$$J = \begin{bmatrix} \frac{\partial x}{\partial \hat{x}} & \frac{\partial x}{\partial \hat{y}} \\ \frac{\partial y}{\partial \hat{x}} & \frac{\partial y}{\partial \hat{y}} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{T}}{\partial u} & \frac{\partial \mathcal{T}}{\partial v} & \frac{\partial \mathcal{T}}{\partial w} \end{bmatrix} \begin{bmatrix} \frac{\partial u}{\partial \hat{x}} & \frac{\partial u}{\partial \hat{y}} \\ \frac{\partial v}{\partial \hat{x}} & \frac{\partial v}{\partial \hat{y}} \\ \frac{\partial w}{\partial \hat{x}} & \frac{\partial w}{\partial \hat{y}} \end{bmatrix},$$

with variable change ($u = 1 - \hat{x} - \hat{y}$, $v = \hat{x}$, $w = \hat{y}$) [8],

$$\begin{bmatrix} \frac{\partial u}{\partial \hat{x}} & \frac{\partial u}{\partial \hat{y}} \\ \frac{\partial v}{\partial \hat{x}} & \frac{\partial v}{\partial \hat{y}} \\ \frac{\partial w}{\partial \hat{x}} & \frac{\partial w}{\partial \hat{y}} \end{bmatrix} = \begin{bmatrix} -1 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix},$$

Table 1: Fifteen control values for $det(J)$ of a cubic triangle

P_{ijk}	Control Value
P_{400}	$9(a_1 \times a_2 \cdot \vec{n})$
P_{040}	$9(b_1 \times b_2 \cdot \vec{n})$
P_{004}	$9(c_1 \times c_2 \cdot \vec{n})$
P_{220}	$\frac{3}{2}(a_1 \times b_2 \cdot \vec{n} + b_1 \times a_2 \cdot \vec{n} + 4e_1 \times e_2 \cdot \vec{n})$
P_{202}	$\frac{3}{2}(a_1 \times c_2 \cdot \vec{n} + c_1 \times a_2 \cdot \vec{n} + 4d_1 \times d_2 \cdot \vec{n})$
P_{022}	$\frac{3}{2}(b_1 \times c_2 \cdot \vec{n} + c_1 \times b_2 \cdot \vec{n} + 4f_1 \times f_2 \cdot \vec{n})$
P_{301}	$\frac{9}{2}(a_1 \times d_2 \cdot \vec{n} + d_1 \times a_2 \cdot \vec{n})$
P_{310}	$\frac{9}{2}(a_1 \times e_2 \cdot \vec{n} + e_1 \times a_2 \cdot \vec{n})$
P_{130}	$\frac{9}{2}(b_1 \times e_2 \cdot \vec{n} + e_1 \times b_2 \cdot \vec{n})$
P_{031}	$\frac{9}{2}(b_1 \times f_2 \cdot \vec{n} + f_1 \times b_2 \cdot \vec{n})$
P_{103}	$\frac{9}{2}(c_1 \times d_2 \cdot \vec{n} + d_1 \times c_2 \cdot \vec{n})$
P_{013}	$\frac{9}{2}(c_1 \times f_2 \cdot \vec{n} + f_1 \times c_2 \cdot \vec{n})$
P_{211}	$\frac{3}{2}(a_1 \times f_2 \cdot \vec{n} + f_1 \times a_2 \cdot \vec{n} + 2d_1 \times e_2 \cdot \vec{n} + 2e_1 \times d_2 \cdot \vec{n})$
P_{121}	$\frac{3}{2}(b_1 \times d_2 \cdot \vec{n} + d_1 \times b_2 \cdot \vec{n} + 2e_1 \times f_2 \cdot \vec{n} + 2f_1 \times e_2 \cdot \vec{n})$
P_{112}	$\frac{3}{2}(c_1 \times e_2 \cdot \vec{n} + e_1 \times c_2 \cdot \vec{n} + 2d_1 \times f_2 \cdot \vec{n} + 2f_1 \times d_2 \cdot \vec{n})$

therefore,

$$J = \begin{bmatrix} \frac{\partial \mathcal{T}}{\partial u} & \frac{\partial \mathcal{T}}{\partial v} & \frac{\partial \mathcal{T}}{\partial w} \end{bmatrix} \begin{bmatrix} -1 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{T}}{\partial v} - \frac{\partial \mathcal{T}}{\partial u} & \frac{\partial \mathcal{T}}{\partial w} - \frac{\partial \mathcal{T}}{\partial u} \end{bmatrix}.$$

Finally,

$$det(J) = (\frac{\partial \mathcal{T}}{\partial v} - \frac{\partial \mathcal{T}}{\partial u}) \times (\frac{\partial \mathcal{T}}{\partial w} - \frac{\partial \mathcal{T}}{\partial u}) \cdot \vec{n},$$

where $\vec{n}$ is the vector $(0, 0, 1)$. Because the derivative of a *qth* order Bézier function is a $(q-1)th$ order Bézier function and the product of two Bézier functions is also a Bézier function, the resulting *Jacobian* is a Bézier polynomial function with order $2(q-1)$. In our case, the *Jacobian* is a fourth order Bézier polynomial with fifteen control points. Specifically,

$$\mathcal{T}^4(u, v, w) = \sum_{i+j+k=4} B^4_{ijk}(u, v, w) P_{ijk},$$

where P_{ijk} is one of the fifteen control values, $B^4_{ijk}(u, v, w) = \frac{4!}{i!j!k!} u^i v^j w^k$, $u \in [0, 1]$, $v \in [0, 1]$ and $w \in [0, 1]$ are the barycentric coordinates and $u + v + w = 1$. Because

$$\frac{\partial \mathcal{T}}{\partial v} - \frac{\partial \mathcal{T}}{\partial u} = 3u^2 a_1 + 3v^2 b_1 + 3w^2 c_1 + 6uwd_1 + 6uve_1 + 6vwf_1$$

and

$$\frac{\partial \mathcal{T}}{\partial w} - \frac{\partial \mathcal{T}}{\partial u} = 3u^2 a_2 + 3v^2 b_2 + 3w^2 c_2 + 6uwd_2 + 6uve_2 + 6vwf_2,$$

where $a_1 = P_{210} - P_{300}$, $b_1 = P_{030} - P_{120}$, $c_1 = P_{012} - P_{102}$, $d_1 = P_{111} - P_{201}$, $e_1 = P_{120} - P_{210}$, $f_1 = P_{021} - P_{111}$, $a_2 = P_{201} - P_{300}$, $b_2 = P_{021} - P_{120}$, $c_2 = P_{003} - P_{102}$, $d_2 = P_{102} - P_{201}$, $e_2 = P_{111} - P_{210}$, $f_2 = P_{012} - P_{111}$, the fifteen control values can be calculated. They are listed in Table 1.

If the element is valid, it means the *Jacobian* is positive everywhere in this element. However, if the computed lower bound of the *Jacobian* is non-positive, it does not necessarily mean that the element is invalid. Since it is only a sufficient condition to calculate a lower bound of the *Jacobian*, sometimes, it is overly conservative. In the cases that the bound is not tight, the minimum value could be positive whereas the element is reported invalid. To further confirm the answer, we obtain the tighter bound by refining the convex hull using the Bézier subdivision algorithm. The algorithm relies on the convex hull property and the de Casteljau algorithm [7].

Indeed, if the negative minimum of the fifteen control values corresponds to one of the vertices of the element, then the element is invalid. If not, and the negative minimum of the fifteen control values corresponds to one of the three nodes on the edge, then it is necessary to refine this edge. We use the Bézier subdivision algorithm to split the edge into two sub-edges. If the negative minimum of the fifteen control values corresponds to one of the three nodes on the face, then it is necessary to refine this face. We use the Bézier subdivision algorithm to split the face into three sub-faces. In this way, the new control polygons are closer to the original polynomial, and the bound becomes much tighter. Other algorithms such as degree elevation could also be used, but the Bézier subdivision algorithm is selected here because the convergence of this repeated subdivision process is very fast [4,5].

It is possible to identify which curved mesh edges need to be corrected when a negative lower bound at a specific control point is found. For example, if the negative lower bound occurs on a mesh vertex, the edges connected to the vertex are the candidates. If one of the two edges is a boundary edge, it is assumed to be correct, then the other one is to be corrected. If the negative lower bound occurs on a mesh edge, then this edge is the one to be corrected. If the negative lower bound occurs on a face, then the edges that bound the face need to be corrected except the boundary edges. Once the edge causes negative *Jacobian* is indentified, local mesh modifications can be used to correct the shape.

Because each curved edge has two control points in the middle, and each control point determines the curvature of the corresponding half of the edge, we first distinguish which part of the curved edge is intersected. For example, if the left half of the curved edge is intersected, that means the curvature determined by the corresponding control point is not large enough. We enlarge the curvature by rotating the segment formed by the left endpoint and the left control point around the left endpoint by a small angle. We do it repeatedly until the lower bound of the *Jacobian* becomes positive.

6. Geometric and topological fidelity to boundaries

While smoothing out contours, the curved edge should not deviate too much from the linear edge to preserve the shape and avoid interference with other curved edges. The algorithm we present in this paper offers a mathematical guarantee that the boundary of the high-order mesh it produces is a faithful representation of the geometric shape within a requested fidelity tolerance. We present proofs of the fact here.

The linear mesh constructed by the method in Sect. 3 provides a faithful representation of the boundary. To measure the distance between the geometric boundaries and the boundaries of the corresponding sub-mesh, we use the two-sided Hausdorff distance. This measure requires that the boundaries of the linear mesh be within the requested tolerance. Below we prove that the curved mesh boundary cannot deviate from the straight mesh boundary by more than a small multiple of the fidelity tolerance, and therefore, for a given value of the fidelity tolerance, we can accommodate both straight and curved deviations. However, the supplied fidelity tolerance must be strictly positive.

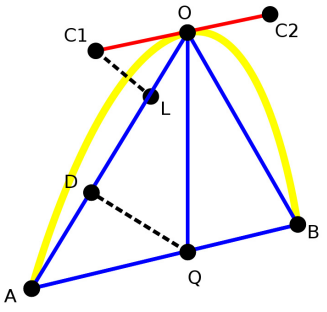


Fig. 4: An illustration of the deviation from the curved edge to the original linear edge. C_1 is a control point of curved edge $\widetilde{AO}$, C_2 is a control point of curved edge $\widetilde{OB}$. Find the point Q such that $|QA|/|QB| = |OA|/|OB|$. $\overline{CL} \perp \overline{OA}$, $\overline{QD} \perp \overline{OA}$.

Table 2: Number of detected invalid elements for the three examples below

Image	Elements	Invalid edges	Corrected edges
SPL brain atlas	3034	1	1
SPL abdominal atlas	2025	1	1
NASA Shuttle	4572	0	0

Since the curved mesh is transformed from the linear mesh, the deviation of the curved edge from the linear edge influences the fidelity. In Fig. 4, let's consider the deviation (say X) of curved edge $A\overset{\frown}{O}$ to linear edge $\overline{AO}$. The length of segment $\overline{C_1C_2}$ controls the curvature of the curve at the Bézier endpoint. Now we need to bound the deviation from the curved edge to the original linear edge. We fix the length of segment $\overline{C_1C_2}$ such that $|\overline{C_1C_2}|$ equals to half of the length of shortest linear edge. Due to the convex hull property, the maximum deviation from the curved edge to the original linear edge is less than the distance from the control point to the linear edge, and then we have

$$X < |C_1L| < |C_1O|.$$

Therefore, the deviation of the curved edge from the linear edge is bounded. The boundary is completely enclosed by the requested tolerance while maintaining the smoothness.

7. Mesh examples

We apply our algorithm to a variety of examples in the following. For these examples, the input data is a two-dimensional image. The procedure described in Section 4.2 was implemented in MATLAB. All the other steps were implemented in C++ for efficiency.

In both of the brain atlas [20] and abdominal atlas [20], the size are $256 * 256$ pixels. Each pixel has side lengths of 0.9375 and 0.9375 units in x, y directions, respectively. The size of the NASA Shuttle is $500 * 350$ pixels, and the pixels have side lengths of 1unit in both x and y dimensions. Table 2 lists the total number of elements, number of actual invalid elements and number of corrected edges in the final meshes of the three examples. Several figures show the result of each step.

8. Conclusion

We presented a new approach for automatically constructing a high-order mesh to represent geometry with smooth boundaries and with guaranteed fidelity and validity. The algorithm we presented is sequential. Our future work includes the development of the corresponding parallel algorithm and the extension to the three-dimensional high-order mesh generation.

9. Acknowledgments

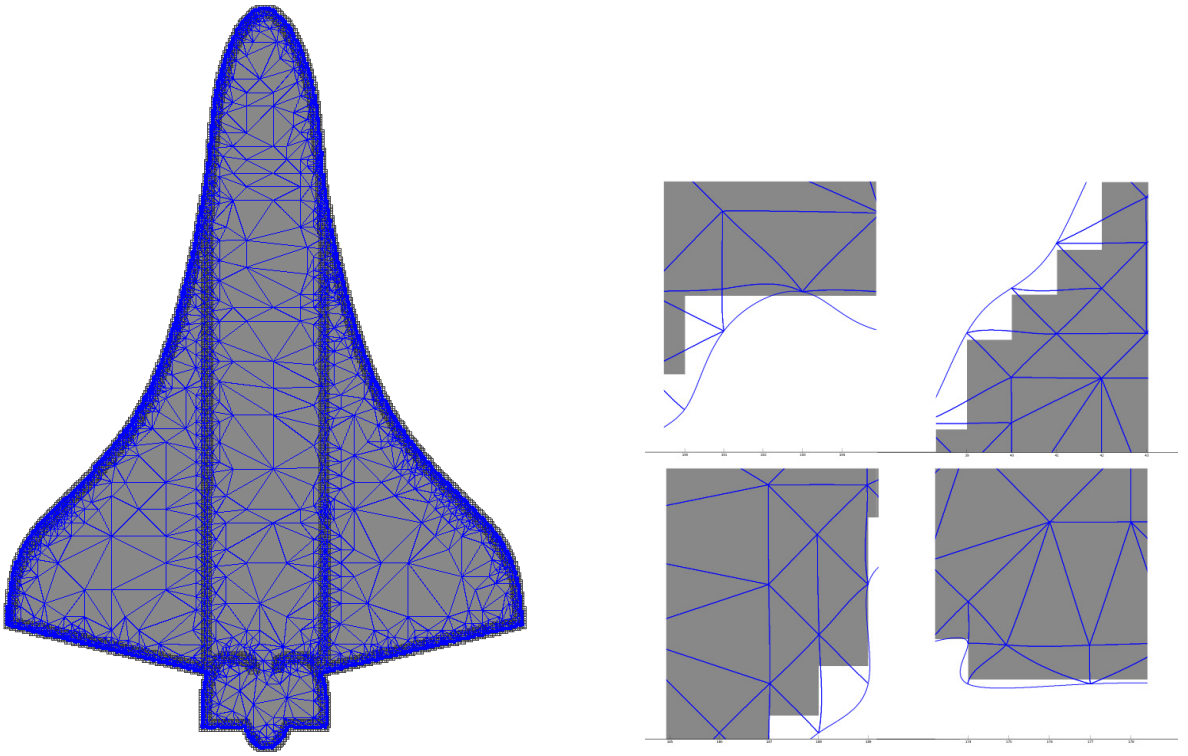
This work was supported (in part) by the Virginia Space Grant Consortium and by the Modeling and Simulation Graduate Research Fellowship Program at the Old Dominion University.

References

[1] I. Babuska and M. Suri. The p and h-p versions of the finite element method, basic priciples and properties. *SIAM J.Numer*, 36:578–631, 1994.

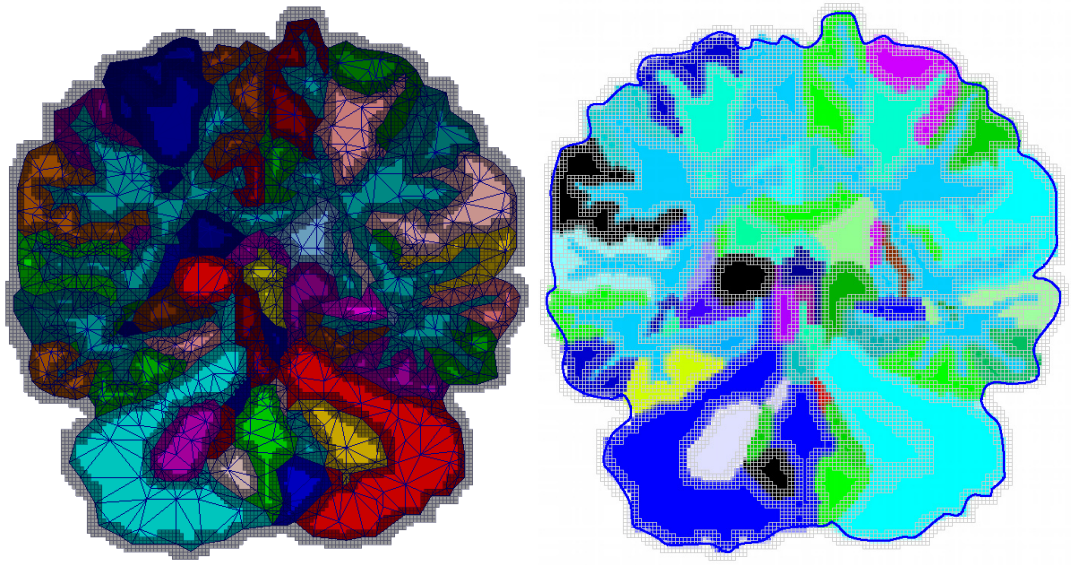
[2] I. Babuska and B. Szabo. Trends and New Problems in Finite Element Methods. *The Mathematics of Finite Elements and Applications*, pages 1–33, 1997.

[3] A. Chernikov and N. Chrisochoides. Multitissue tetrahedral image-to-mesh conversion with guaranteed quality and fidelity. *SIAM Journal on Scientific Computing*, 33:3491–3508, 2011.



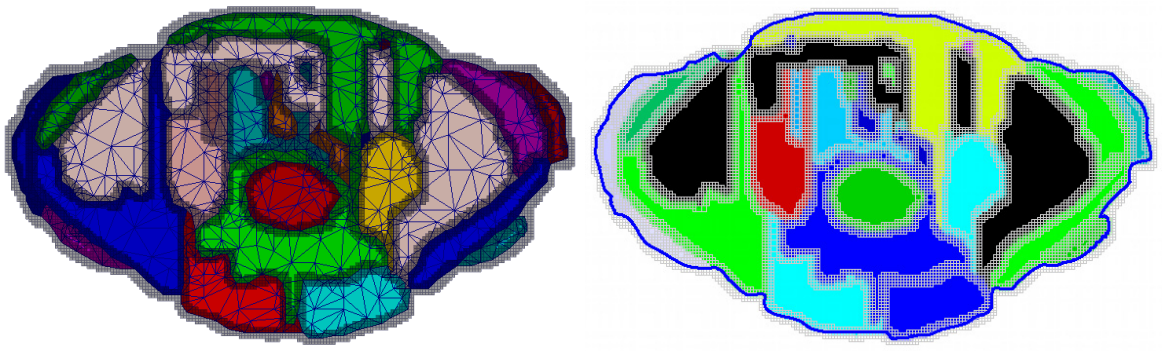
(a) (b)

Fig. 5: (a) Curved mesh of a NASA Shuttle with smooth boundary. (b) Some zoomed-in parts of the curved mesh.



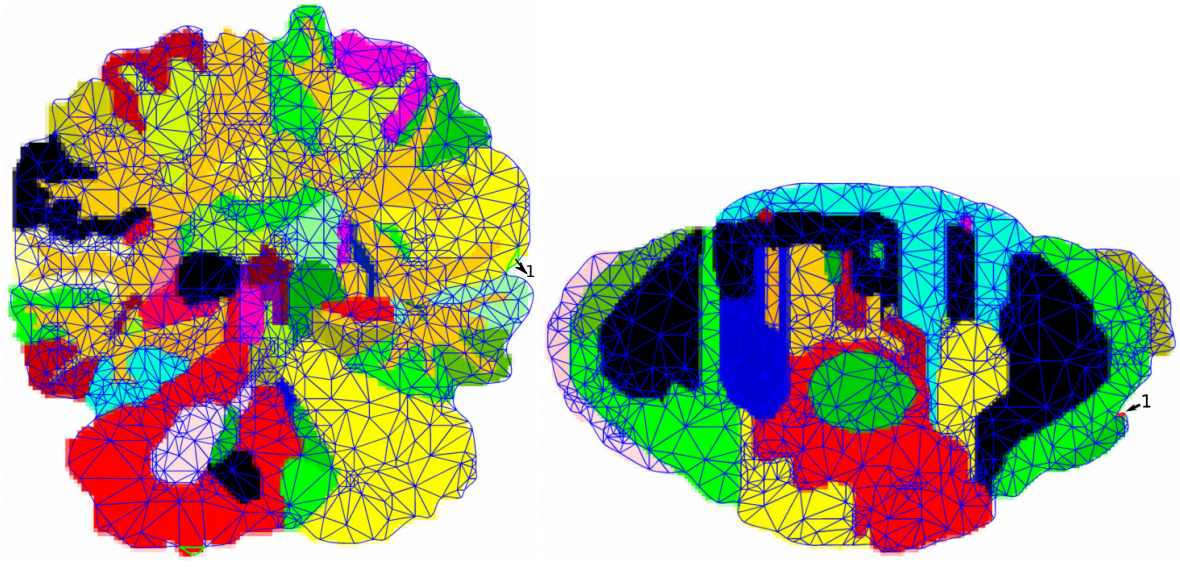
(a) (b)

Fig. 6: Results of the first two steps of our algorithm. (a) Linear mesh of a slice of the brain atlas within two pixels fidelity tolerance. (b) Smooth curved boundary of a slice of the brain atlas within two pixels fidelity tolerance.



(a) (b)

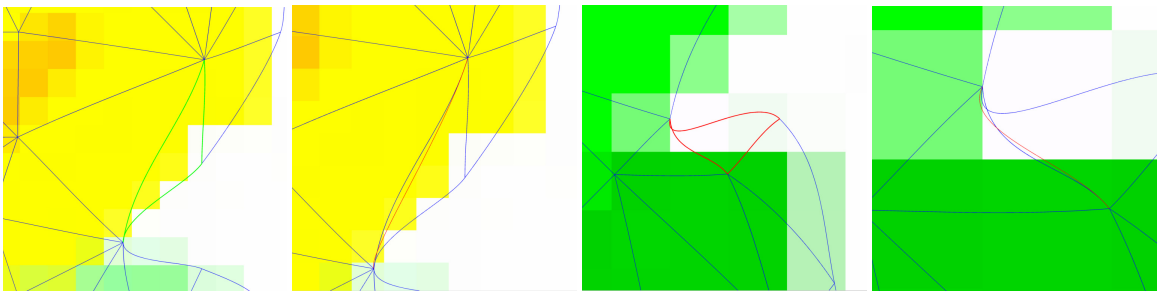
Fig. 7: Results of the first two steps of our algorithm. (a) Linear mesh of a slice of the abdominal atlas within two pixels fidelity tolerance. (b) Smooth curved boundary of a slice of the abdominal atlas within two pixels fidelity tolerance.



(a) (b)

Fig. 8: Results of curving mesh entities in the interior according to the equilibrium solution of a linear elasticity problem. (a) After calculating the lower bound of the *Jacobian* for each element, one was reported having the non-positive value. The invalid element is highlighted in green bold curves with a number. (b) The invalid element is highlighted in red bold curves with a number.

[4] E. Cohen and L. Schumaker. Rates of convergence of control polygons. *Computer Aided Geometric Design*, 2:229–235, 1985.
[5] W. Dahmen. Subdivision algorithm converge quadratically. *J. of Computational and Applied Mathematics*, 16:145–158, 1986.
[6] S. Dey, M. S. Shephard, and J. E. Flaherty. Geometry representation issues associated with p-version finite element computations. *Computer Methods in Applied Mechanics and Engineering*, 150:39–55, 1997.
[7] G. Farin. *Curves and Surfaces for Computer-Aided Geometric Design*. Academic Press, 1997.
[8] P. L. George and H. Borouchaki. Construction of tetrahedral meshes of degree two. *Int. J. Numer. Mesh. Engng*, 90:1156–1182, 2012.
[9] A. Johnen, J. F. Remacle, and C. Geuzaine. Geometrical validity of curvilinear finite elements. *Journal of Computational Physics*, 233:359–372, 2013.
[10] G. Karniadakis and S. J. Sherwin. *Spectral/hp Element Methods for CFD, second edition*. Oxford Univerisity Press, Great Clarendon Street,



(a) (b) (c) (d)

Fig. 9: After the detection of the invalid elements, the checking procedure reported the intersected mesh edges and correct them accordingly. (a) A zoomed-in view of an invalid element, high-lighted by green bold curves. (b) A zoomed-in view of the intersected mesh edge corrected by the red curve. (c) A zoomed-in view of an invalid element, high-lighted by red bold curves. (d) A zoomed-in view of the intersected mesh edge corrected by the red curve.

Oxford, 2004.

[11] C. G. Kim and M. Suri. On the p-version of the finite element method in the presence of numerical integration. *Numer. Methods*, 9:593–629, 1993.

[12] L. Liu, Y. J. Zhang, T. J. R. Hughes, M. A. Scott, and T. W. Sederberg. Volumetric T-spline construction using Boolean operations. *Engineering with Computers*, 2014.

[13] Q. K. Lu, M. S. Shephard, S. Tendulkar, and M. W. Beall. Parallel mesh adaptation for high-order finite element methods with curved element geometry. *Engineering with Computers*, 30:271–286, 2014.

[14] X. J. Luo, M. S. Shephard, L. Q. Lee, L. Ge, and C. Ng. Moving curved mesh adaptation for high-order finite element simulations. *Engineering with Computers*, 27:41–50, 2011.

[15] X. J. Luo, M. S. Shephard, R. M. O'Bara, R. Nastasia, and M. W. Beall. Automatic p-version mesh generation for curved domains. *Engineering with Computers*, 20:273–285, 2004.

[16] P. O. Persson and J. Peraire. Curved Mesh Generation and Mesh Refinement using Lagrangian Solid Mechanics. In *Proceedings of the 47th AIAA Aerospace Sciences Meeting and Exhibit*, Orlando, FL, January 2009.

[17] O. Sahni, X. J. Luo, K. E. Jansen, and M. S. Shephard. Curved boundary layer meshing for adaptive viscous flow simulations. *Finite Elements in Analysis and Design*, 46:132–139, 2010.

[18] M. S. Shephard, J. E. Flaherty, and K. E. Kenneth E. Jansen. Adaptive mesh generation for curved domains. *Applied Numerical Mathematics*, 52:251–271, 2005.

[19] S. J. Sherwin and J. Peiro. Mesh generation in curvilinear domains using high-order elements. *Int. J. Numer*, 00:1–6, 2000.

[20] I. Talos, M. Jakab, R. Kikinis, and M. Shenton. Spl-pnl brain atlas, 2008.

[21] O. C. Zienkiewicz, R. L. Taylor, and J. Z. Zhu. *The Finite Element Method: Its Basis and Fundamentals, 6th edition*. Oxford: Butterworth-Heinemann, 2005.